Journal of Ubiquitous Systems and Pervasive Networks
Volume 2, No. 1 (2011) pp. 1-13

A Generic Autonomic Architecture To Improve Routing In MANETs

Anis Ben Arbia [a*], Habib Youssef [b]

[a] *PRINCE Research Unit H. Sousse TUNISIA, anis.benarbia@infcom.rnu.tn*
[b] *PRINCE Research Unit H. Sousse TUNISIA, Habib.youssef@fsm.rnu.tn*

Abstract
Wireless ad-hoc networks are highly dynamic networks for which routing is a challenging task. Existing routing protocols whether reactive, proactive or hybrid showed limitations in various situations. To optimize the behavior of the routing nodes, the academic community is recommending the reliance on an autonomic approach that would guarantee self–adaptation, self–configuration, self–management, self-optimization and self–protection mechanisms of all the network nodes. In this paper we propose an autonomic architecture geared towards improving the routing performance within wireless ad-hoc networks. This architecture provides a formal framework where each node has a routing agent whose role is to monitor routing activities within the network, which are used to tune the routing decisions of the node, in harmony with other nodes. The architecture is easy to tailor to any routing protocol. Simulation results with autonomic implementations of AODV and OLSR routing protocols show remarkable performance improvement with respect to efficiency (Packet Delivery Ratio, overhead, quality of paths and reliability).
Keywords: *Wireless Ad-hoc Networks, Routing, Autonomic Architecture, AODV, OLSR.*

1. Introduction

Modern networks are highly complex systems, mixing a variety of communication technologies and offering a wide myriad of services. Human capabilities are outpaced by difficulties imposed by the complications of emerging communications technologies and services. Nowadays autonomic architectures offer a suitable solution to the intelligent operations of such networks. An autonomic system is Self–adaptable, Self–manageable, Self-Configurable and Self-protectable.

Routing in wireless ad hoc networks presents one of these tremendous challenges due to the dynamic nature of the network caused by the lack of a stable routing infrastructure. Numerous routing protocols have been proposed in the literature and some of which have been ratified. All exhibit inefficiencies under certain network conditions. Researchers started recently to recommend adaptive versions of these routing protocols [1], [2], [3] (see §. 2).

The novelty of this work consists of proposing an autonomic architecture geared toward supporting routing activities within wireless ad-hoc networks. It is a distributed architecture composed of communicating Routing Learning Agents (RLAs), one per node. The RLAs listen promiscuously to the routing traffic (requests, responses, and errors) exchanged by the nodes. RLAs consider the captured traffic, over the time, as events and construct an overview of the relationships between nodes involved in these events. Indeed, based on observed routing traffic, each RLA scores the nodes using entropy based approach. Each RLA uses the entropy scores to classify the nodes behaviors to be either, (1) inhibiting, (2) negligent, (3) uncooperative, (4) cooperative, (5) diligent, or (6) activator, listed from the least to the most cooperative. As we shall see, such classification helps noticeably improve the quality of established routes. Routing overhead will also be reduced since better routes result in lower route failures and re-routing requests. Thus, in our approach, each RLA is capable of analyzing the behavior of known nodes and to dynamically adapt its routing decisions. It is able to observe, analyze, and intelligently intervene on the local node routing table. Routing activities observed and used by an RLA depend on the routing protocol employed. For example, for the AODV protocol (Ad-hoc On demand Distance Vector), nodes can learn useful network state information (topology and workload) from observed Route Requests, Route Replies and Route Errors. For OLSR protocol [4], [16] HELLO messages and TC (Topology Control) messages are also useful information.

RLAs' architectural concept consists of three actors:

* Corresponding author. Tel.: +21625631500
Fax: +21673364411; E-mail: anis.benarbia@infcom.rnu.tn

DOI: 10.5383/JUSPN.02.01.001

Autonomic Manager (AM), Touch Points (TP) and Managed Resources (MR):

--Autonomic Manager (AM) is the core of the architecture, as it ensures the system self-managing property. The AM is composed by four modules: Monitor, Analyzer, Planner and Executor.

--Touch Points (TP): TP are the interfaces of the AM to the system environment. It is composed of two components: effectors and sensors. Sensors offer a set of logical parameters (metrics) describing a real time state of the current configuration of the system. Effectors are responsible of manipulating MR. So all updates (new configurations and notifications) are applied to the MR through the effectors. We note that TP (Sensors) can use a tracing technique (gathering observations from a given execution) to detect problems involved in the application.

-- Managed Resources (MR): these present a set of configurable parameters defined on the basis of specific system needs. These parameters can be attributes, errors, functionality symptoms, alert degrees, abnormal response time, etc.

Hence, an autonomic manager should be able to observe (using Sensors), understand observations by using appropriate metrics (Monitor), analyzes the current state of the system and proposes specific reactions that resolve the problem (Analyzer), create or invoke methods to act (Planner), and finally apply required changes on managed resources. As shown in Figure 1, the proposed autonomic architecture is consolidated by a Local Knowledge Base (LKB), which serves to calibrate decisions of the local Autonomic Manager. In addition, the LKB provides useful data for the analyzer.

The rest of this paper is organized as follows. First we describe related work. Second we present the behavioral model. Then we provide a detailed description of the RLA architecture. Next the fundamental properties of the proposed architecture are presented. In the following section we present and discuss simulation results comparing the performance of AODV and OLSR routing protocols with that of their autonomic implementations, namely A^{2}ODV and AOLSR. Finally we conclude the paper.

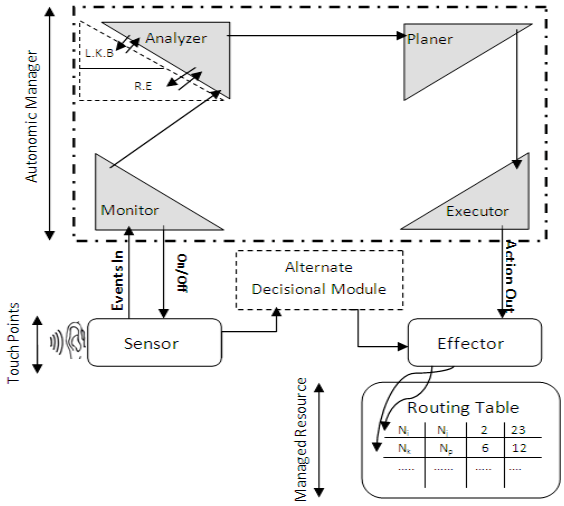

Fig. 1. RLA Architecture. L.K.B: Local Knowledge Base, R.E: Rules Engine, ADM: Alternate Decisional Module

2. Related Works

Many leading wireless routing protocols adopted by IETF, such as AODV [5] and OLSR [4] exhibit serious performance degradation due to the characteristics of this type of networks. For several years, researchers focused on the optimization of existing protocols by trading off QoS vs. overhead [6], [7], [8]; [9], energy consumption vs. routing quality [10] routing overhead vs. QoS, etc. Other previous work has proposed a wireless routing protocol using an adaptive solution, which can guarantee an adaptive behavior of the protocol. Some of these works propose adaptive versions which are based on monitoring processes. The authors in [2] propose that each node has to monitor the current congestion status. If congestion is detected then an alternate path has to be proposed. In that case congestion at the node will be reduced compared to the originally established path. The overhead caused by the selection of the alternate path is unacceptable and induces a large number of control packets. Another work called AntNet proposed by [11] in which routing operations are performed based on the gathering of useful information. AntNet operates with two types of network exploration: forward and backward. The forward exploration observes changes and the backward performs required updates in the routing table. The authors of [12] propose autonomic and decentralized management architecture for MANETs, which deploys dynamic loading policies. At each transaction, the autonomic manager updates required policies and redistributes them, which ensures a continuous control and enables a dynamic management closely related to the current state of the network.

IBM was among the first to conduct research on Autonomic systems by offering a reference autonomic architecture [17]. As shown in Figure 2, the IBM architecture presents general concepts which transform a given classic system in an autonomic one. IBM provides three actors: Autonomic Manager (AM), Touch Points (TP), Managed Resources (MR) and a Local knowledge Base (L.K.B) linked to all other modules of an AM. Table. 1 compares our architecture with that of IBM. With respect to the IBM architecture [13], [17], we added an Alternate Decisional Module (ADM) permitting the treatment of urgent cases without the intervention of the AM. This can be particularly helpful in highly dynamic environments. Further, we have equipped the Analyzer with a Rules Engine (RE) enabling the node to intelligently adapt its behavior to observed routing traffic. The RE relies on six rules, (see next section) serving to detect nodes attitudes or behaviors towards each others. Finally, the Local Knowledge Base (LKB) is linked only to the Analyzer, so that read/write actions can only be performed by the analyzer.

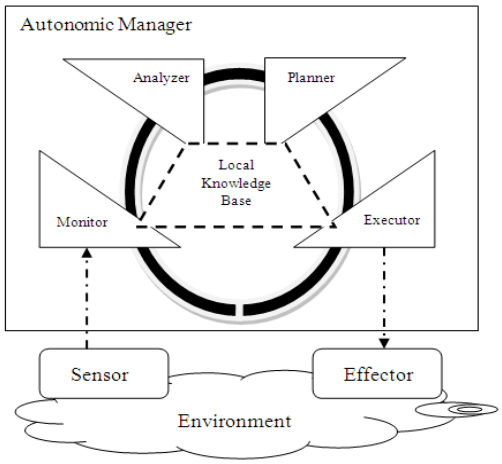

Fig. 2. IBM Autonomic Architecture reference model

Table 1. Comparison between the RLA and the IBM architectures

Module	RLA	IBM
Sensor	Traffic observatory	A probe linked to the system
Effector	Routing updates	Updates Applicators
Monitor	Yes	Yes
Analyzer	Yes	Yes
Planner	Yes	Yes
Executor	Yes	Yes
ADM	Yes: treatment of urgent cases	No
LKB	Yes: Linked only to the analyzer	Yes: Linked to all other modules of the AM
RE	Yes: located and managed by the analyzer	No

3. Behavioral Analysis

RLA is based on a Behavioral Analysis (BA) approach. The BA provides the means of analyzing and understanding behaviors of entities based on observed events. Several researchers have reported that behaviors can be learned from observed interactions between entities themselves and/or their environment. That is, the behavior of each entity towards others can be understood based on its past and present interactions with other entities.

Behavioral Analysis is on the crossroad of several research fields like Social and Behavioral Sciences (Economics, Psychology, and Sociology), Artificial Intelligence (Intelligent Agents, Multi-agent systems), etc. For example, in economics model, researchers use BA in order to study the behavior of decisions makers' attitudes (optimism, pessimism, speculation, caution, imitation, leadership, etc.)

In the case of ad hoc networks, a mobile node behaves according to its own needs, its state, and its current perception of the network. Thus, a node behavior closely depends on the needs and state of other nodes in the network, since they act as routers for each others. The quality of established routes can be noticeably improved if the routing process is behavior aware. For these reasons, each node in the wireless ad hoc network (via its RLA) is equipped with the capability of capturing the behavior of other nodes based on observed routing events (route requests, responses and errors), which it uses to infer the attitude of each node towards other nodes.

Numerous simulations with various scenarios have demonstrated that a node can take one the following routing attitudes:

1. Inhibitor: a node which always replies with errors to routing solicitations of all nodes, thus inhibiting traffic from the network.
2. Negligent: a node which replies negatively to all routing solicitations issued by this local node.
3. Uncooperative: a node which positively replies to routing requests only when it is in its own interests to intervene. That is, the node is helpful only to those nodes that appear in its currently known active routes.
4. Cooperative: a node which always positively replies to routing solicitations from the local node with its best route.
5. Diligent: a node which always positively replies to routing solicitations from any node with its best route.
6. Activator: a node which never replies with errors and it offers positive responses for all routing needs; in addition an activator node automatically propagates all route updates to other nodes that are activator to him. Note that if a node i is activator to a node j, then node j must also be activator to node i.
7. Unknown: this is also the initial state of a node. A node can also be classified as having an unknown attitude if it keeps rapidly changing its attitude, which makes it impossible to properly classify it.

Table 2 summarizes simulations based on 20 scenarios which involve extensive routing interactions between 100 mobile nodes. These simulations confirm that the above attitude classification exhaustively cover all possible node routing behaviors. Indeed, as shown in Table 2, the time distribution of the various attitude states sum up to 1. Further, a node spends most of its time in one of the known attitude states and two nearby states. A close look at Table 2 allows us to conclude the following:

- If a node has a given attitude, the time percentage it sojourns at that state is largest compared to sojourn time percentages at other attitudes (blue box in Table 2).
- The light gray boxes in Table 2 show that the adjacent attitude states have the next largest sojourn time percentages.
- The dark gray boxes show that a node in an unknown attitude sate can change to all other attitude states, practically with the same percentage (dark gray boxes in Table 2).

In our proposal, the RLA of each node has the responsibility of performing a behavioral analysis of other communicating nodes and classifying their routing attitudes, based on routing traffic captured by the local node. Routing traffic consists of routing events which can be a NRE (New Route Event, e.g. Route Request or Route Reply in AODV protocol) or an RFE (Route Failure Event, e.g. Route Error in AODV protocol). The first type presents favorable interactions between nodes while the second type represents unfavorable ones. For example, a node J which issues only RFE events to node I will be considered as Inhibitor for I. By contrast, a node J which always issues NRE events to I (and vice versa) will be considered as activator.

To permit a node determine behavioral attitudes of other interacting nodes, we have defined an events model which is used by the node RLA agent (Figure 1: Rule Engine). In the following we present the model rules, whose verifications are triggered at the occurrence of each routing event.

Let T be a period of time, and e an event which can be NRE or RFE. Assume that during T, N_i events have occurred at a particular local node i.

Let E_i = {e, where e is an event received or issued by the local node i during T}; then Cardinal (E_i) = N_i. We denote by $RFE_{j\rightarrow i}$ (respectively $NRE_{j\rightarrow i}$) an RFE event sent by node j to node i (respectively a NRE event sent by node j to i). Then the various routing attitudes of a particular node j can be expressed as follows:

Table 2. Attitudes variations during simulations

↱Changes attitudes to	Inhibitor	Negligent	Uncooperative	Cooperative	Diligent	Activator	Unknown
Inhibitor	41%	29%	16%	7%	5%	1%	1%
Negligent	25%	39%	22%	6%	5%	2%	1%
Uncooperative	8%	22%	37%	23%	7%	2%	1%
Cooperative	2%	3%	23%	37%	22%	12%	1%
Diligent	3%	3%	10%	24%	36%	23%	1%
Activator	2%	3%	7%	16%	19%	52%	1%
Unknown	14%	15%	16%	13%	16%	16%	10%

A node j is considered as inhibitor towards a node i if it satisfies R1:

R1. ($\forall\ e \in E_i$ where j is involved then $e = \mathrm{RFE}_{j\to i}$)

A node j is considered as negligent towards a node i if it satisfies R2:

R2. If $\exists\ e = \mathrm{NRE}_{i\to j}$ then $\nexists\ e \in E_i$ where $e = \mathrm{NRE}_{j\to i}$

A node j is considered as uncooperative towards a node i if it satisfies R3:

R3. (If $\exists\ e = \mathrm{NRE}_{i\to j}$ then $\nexists\ e \in E_i$ where $e = \mathrm{NRE}_{j\to i}$) AND ($\forall\ e \in E_i,\ e \neq \mathrm{RFE}_{j\to i}$)

A node j is considered as cooperative towards a node i if it satisfies R4:

R4. If $\exists\ e = \mathrm{NRE}_{i\to j}$ then $\exists\ e \in E_i$ where $e = \mathrm{NRE}_{j\to i}$

A node j is considered as diligent towards a node i if it satisfies R5:

R5. (If $\exists\ e = \mathrm{NRE}_{i\to j}$ then $\exists\ e \in E_i$ where $e = \mathrm{NRE}_{j\to i}$) AND ($\forall\ e \in E_i,\ e \neq \mathrm{RFE}_{j\to i}$)

A node j is considered as activator towards a node i if it satisfies R6:

R6. (If $\exists\ e = \mathrm{NRE}_{i\to j}$ then $\exists\ e \in E_i$ where $e = \mathrm{NRE}_{j\to i}$) AND (If $\exists\ e = \mathrm{NRE}_{j\to i}$ then $\exists\ e \in E_i$ where $e = \mathrm{NRE}_{i\to j}$) AND ($\forall\ e \in E_i,\ e \neq \mathrm{RFE}_{j\to i}$)

The above six rules are verified by the rule engine of the RLA at the occurrence of each new routing event.

Now, for a particular node i, and for N_i received events during a period T, how many times a rule must evaluate to true to decide the routing attitude of a particular interacting node j towards node i. This question can be approached by the use of the Shannon's entropy theory. The entropy is an evaluation of uncertainty of a random variable. It can also be considered as a measure of the complexity of a given system (or a degree of its randomness). In statistics, the entropy is used to quantify the value of information of a random variable (quantity which makes it possible to make a decision). According to entropy statistics, the value of information is a function of probability, which takes the following formula:

$$H(X) = -\sum_x p(x)\log_2 p(x) \qquad (1)$$

Where X is a discrete random variable, $x \in X$, and $p(x) = \Pr\{X = x\}$.

Entropy is used to characterize the impurity of a random collection of events. In our case, entropy is used to measure the homogeneity of the events that a rule matches. For a given collection E, containing the events that a certain rule R_i matches, let P_i be the proportion of events in E where the rule R_i evaluates to true ($P(R_i=true)$).

In our case, the RLA of a given local node, after observing N routing events, must identify the routing attitude of every node involved in these routing events. This can be achieved with the help of the entropy function given by Equation (1) as follows. Recall that each attitude is expressed by one of the rules R1 to R6. Then, the entropy associated with each rule R_i can be expressed as follows:

$$H(R_i) = -\sum_{E=\{e:event\}} p(R_i = true)\log_2 p(R_i = true) \qquad (2)$$

Where the probability that a rule R_i evaluates to true is given by the following expression:

$$p(R_i = true) = \frac{k_i}{N} \qquad (3)$$

Where N is the total number of events and k_i is the number of events where $R_i = true$.

Figures 3 to 8 show, respectively, the entropies statistics of rules R1 to R6.

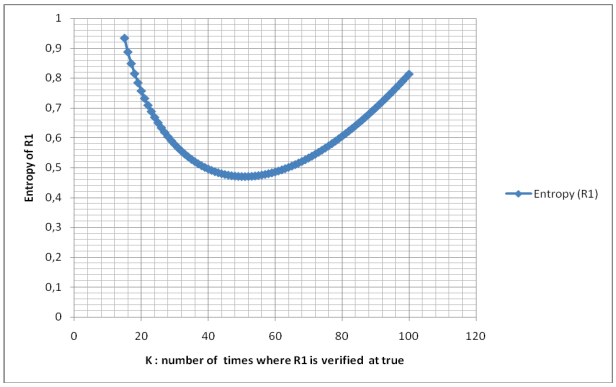

Fig. 3. Entropy for Rule R_1; it is minimum for $k_1 = \lfloor N/2 \rfloor = 50$.

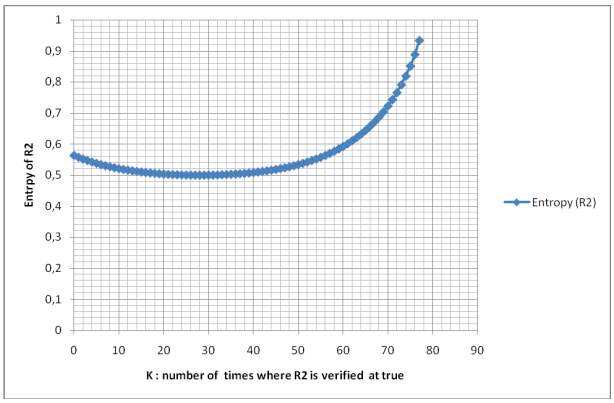

Fig. 4. Entropy for Rule R_2; it is minimum for $k_2 = \lfloor N/22 \rfloor = 25$.

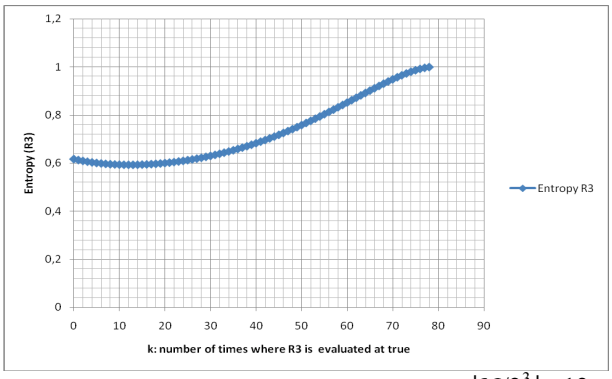

Fig. 5. Entropy for Rule R_3; it is minimum for $k_3 = \lfloor N/2^3 \rfloor = 12$.

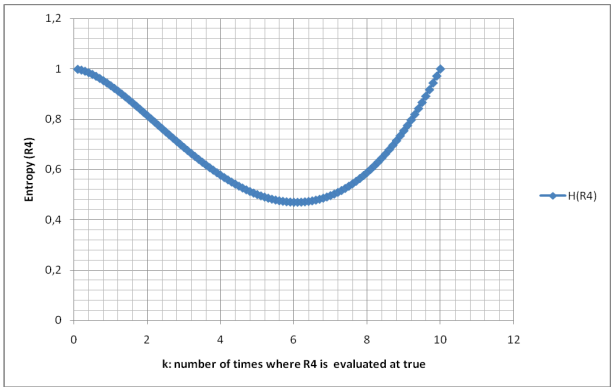

Fig. 6. Entropy for Rule R_4; it is minimum for $k_4 = \lfloor N/24 \rfloor = 6$.

Fig. 7. Entropy for Rule R_5; it is minimum for $k_5 = \lfloor N/25 \rfloor = 3$.

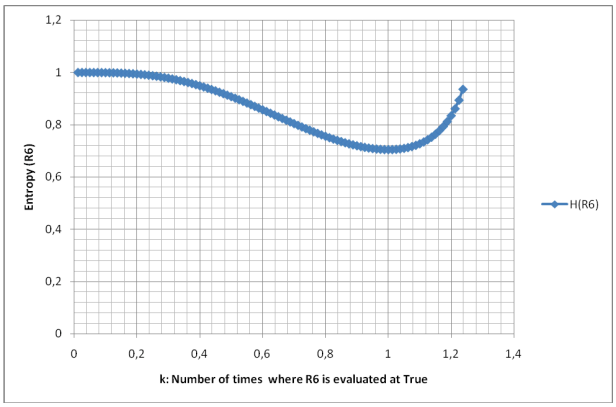

Fig. 8. Entropy for Rule R_6; it is minimum for $k_6 = \lfloor N/26 \rfloor = 1$.

Figures 3 to 8 also show, for each rule, the number of events at which the entropy is minimum. Recall that the entropy is used to identify the value of information which makes it possible to make a decision as to the attitude of a node toward another node: this is when the entropy is at its minimum. So, these values reflect the number of times where a rule has to be evaluated at true to conclude the related attitude.

Table 3. Number of events for which a rule must evaluate to true in order for a node to assume the corresponding attitude

	Attitude	*K*
Rule 1	Inhibitor	$\lfloor N/2 \rfloor$
Rule 2	Negligent	$\lfloor N/2^2 \rfloor$
Rule 3	Uncooperative	$\lfloor N/2^3 \rfloor$
Rule 4	Cooperative	$\lfloor N/2^4 \rfloor$
Rule 5	Diligent	$\lfloor N/2^5 \rfloor$
Rule 6	Activator	$\lfloor N/2^6 \rfloor$

Therefore, an important question is: How many times a rule has to evaluate to true if N_i events are received by node i during a period T, to decide that a node j has a given attitude towards i? Table 3 tells us that Rule 1 must evaluate to True for 50% of the received events for the corresponding node to be assumed Inhibitor; It is 25% of the received events for Rule 2; etc. In general, for a given two nodes i and j, and for N_i received events, a rule R_p has to evaluate to true at least $N/2^p$, $(1 \leq p \leq 6)$ times to decide that the node j has the behavior attitude stated by R_p towards i. That is, Rule i must be true for twice the number of events of Rule i+1, i=1,..,5. This can be explained by the fact that each rule imposes one or more additional conditions on those of the previous rule. Next, we describe the RLA architecture.

4. RLA Architecture

An RLA is composed of three parts: Autonomic Manager, Touch Points, and Managed Resources. In the following we describe each part.

4.1. RLA Managed Resources

In our case, only one managed resource is considered: the routing table. Each RLA has to control the local routing table in order to perform the best possible configuration. The RLA

uses the two fundamental fields which exist in almost all routing tables: Destination Address and Next hop. The RLA has the responsibility of automatically filling these two entries in the routing. The effector can access and manipulate the routing table only after receiving an action request from the AM module.

4.2 RLA Touch Points

The touch points are composed of Sensor and Effector:

- Sensor (Figure 8)

It is the module that implements the observer component. It captures all requests listened by the local node and transforms them into events (Events_In). Requests can be: Route requests, Route responses, Route failures or errors.

The sensor has four interfaces: two IN and two OUT. The two IN, where the first one serves as a listener of events, while the second one is connected to the monitor, which can shut off the sensor using this interface. The two OUT are connected to the Monitor and to the ADM. Events collected by the sensor are filtered by ''the events filter'' which decides about the relevance of each event and sends them to the Monitor or to the ADM. The following algorithm describes the ''Events Filter'' functionalities:

Algorithm Events_Filter

N_i, N_j : Nodes Id

Events_In (N_i, N_j) //Reception by N_i of an event involving N_j

1. *Begin*
2. *If Events_In (N_i, N_j) is urgent and requires ADM*
   2.1 Then send Events_In (N_i, N_j) through OUT_2 // To the Alternate Decisional Module
   2.2 Else send Events_In (N_i, N_j) through OUT_1 // To the Monitor
3. *fi*
4. *End*

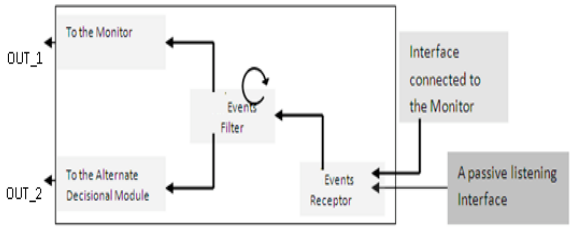

Fig. 8. Structre of a Sensor.

- Effector (Figure 9)

As aforementioned, after receiving events then the AM has to decide whether or not an update will be required. If that is the case, updates have to be applied to the MR using the effector. The effector has three interfaces (Figure 9): two IN and one OUT. The two IN are:

a- IN_1 connected to the executor (which is used to receive ''actions'' from the AM)

b- IN_2 connected to the ADM (which is used to receive ''urgent actions'' from the ADM).

The two interfaces IN are collected in the Actions Receptor, which is responsible for actions synchronization. The synchronization is performed on the basis of the time capture of each event.

The interface OUT is connected directly to the MR (Managed Resource). Three actions are possible according to a specific score attributed to each entry by the analyzer (PS: Pairewise Score, will be defined later in this paper):

1. Delete the Entry (abbreviation: DEL_ENT): if the routing entry is no longer needed, then it can be deleted.
2. Suspend the Entry (abbreviation: SUSP_ENT): if some routing problems (errors, repeated disconnections, etc.) are detected in a given entry then it has to be suspended. In that case, the RLA provides three actions to apply:
   (a) Suspended until a notification from the analyzer: the entry has to be suspended until finding a new route.
   (b) Suspended until next update: if a new entry cannot be found then the entry has to be unsuspended in order to be used at the next routing operation. In that case, the RLA has tried to find a new entry, but there is no new entry available, then it uses the existing one.
   (c) Suspended until a next update of that entry: the entry must remain suspended until a new entry has to be found.
3. Update the Entry (abbreviation: UPD_ENT): the entry has to be updated immediately since a new route is found.

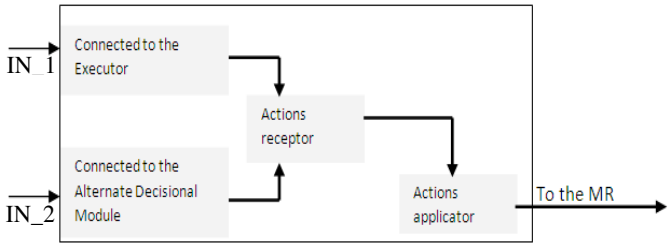

Fig. 9. Structure of an effector.

4.3 RLA Autonomic Manager

The autonomic manager is composed of four modules: Monitor, Analyzer, Planner and Executor:

- Monitor (Figure 10)

The monitor has one interface IN and two interfaces OUT. The IN interface is connected to the sensor and used to receive captured events. The two OUT interfaces are:

(a) OUT_1 connected to the Sensor (which can be used to send requests to the sensor: On / Off capturing)

(b) OUT_2 connected to the Analyzer (which is used to send reports achieved by the report Creator).

The Monitor is composed of: (1) a collector of events, which allows the reception of events from the sensor, (2) an events filter responsible of the crucial selection of events; it classifies received events according to their priorities. Indeed, a node hears on his wireless channel a variety of routing events, many of which may neither concern him nor his immediate neighbors.

The highest priority will be given to events in which the local node is involved such events concern one or more entries in the routing table, thus the correspondent RLA has to treat these events with highest priority. As Medium priority, the monitor distinguishes events which concern immediate neighbors of the local node since neighbor nodes are used as the first interface (hop) to the rest of the network. Thus events involving these nodes have to be considered with a medium priority. Otherwise

events can involve nodes which are:

(a) N-hop neighbors of the local node. These events are treated with low priority since they cannot directly affect the local routing activity. For the local node, these events serve to update the PS score of nodes involved in the events.

(b) Nodes unknown to the local node. In general, these events are ignored.

To summarize, events received by a node are classified as,

H: High, this corresponds to events where the local node is involved.

M: Medium, this corresponds to events where the one-hop neighbors of the local node are involved.

L: Low, this corresponds to events where neither the local node nor its neighbors are involved.

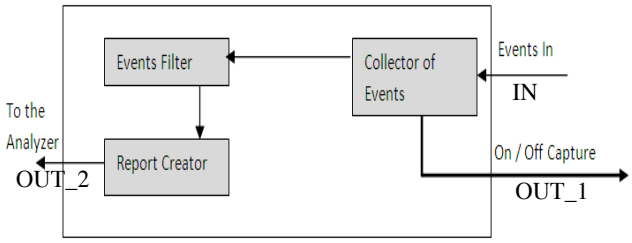

Fig. 10. Structre of a Sensor.

Finally, (3) the third component of the monitor is the report creator, which prepares and sends a detailed report to the Analyzer. Figure 11 shows an example of a report sent by the monitor to the analyzer. The first column shows the number of events collected for each operation; the second column shows the id of the node initiator of the event, concatenated to the type of the event and to the id of the node receptor of the event. The third column indicates the type of the operation (receive or send) concatenated to the time of the capture. Finally, the fourth column gives the assigned priority for each operation.

Fig. 11. Example of a report sent from the Monitor to the Analyzer

2	1 RREQ 9	s 38.194124658	H
5	1 RERR 10	s 39.456821554	H
4	3 RREP 14	r 51.125894565	L
6	5 RERR 2	s 63.845965145	M

- Analyzer (Figure 12)

The Analyzer consists of:

-- An Events receptor,

-- A Controller which hosts a rules engine and connected to the Local Knowledge Base (L.K.B),

-- A Report creator connected to the planner.

The ''Events receptor'' receives messages from the Monitor; it rearranges them following the indicated priorities and sends the obtained results to the controller. At this stage, the controller has a crucial role: it determines correlation between new events (received from the receptor) and old events (stored in the LKB). This correlation can be obtained by looking for the tendency of the relationships between nodes involved in the events list. For this reason, the controller uses the ''Rules Engine'' to discover interactions between nodes based on captured routing traffic (errors, replies, requests…). The ''Rules Engine'' classifies events as: NRE (New Route Events) or RFE (Route Failure Events).

The ''Rules Engine'' operates as defined in ''events model'' section. In addition, at the occurrence of a routing event, the rule engine re-verifies the veracity of each affected rule and passes on the results to the controller. Then, the controller (Figure 13) uses ''Tendency Detection Algorithm (TDA)'' permitting the computation of a score ''PS'' (Pairewise Score) for each node involved in the events list regarding its attitude with the local node. In practice, the TDA determines PS scores based on an incremental counter attached to each known node. The following algorithm describes how the PS scores are computed.

1. *Algorithm PS_Comp (Rule_at_True, N_L, N_j)*
2. *N_L, N_j : Nodes Id*
3. *NE: total number of times Rule_at_True has been evaluated until now*
4. *Begin*

//Increment Counter of the rule verified at true

5. *For each Rule i (i=1 .. 6)do*
6. ***If Counter_of (Rule_at_True, N_L, N_j)≥NE/2^i //verifying Entropy Values***
7. *Then Counter_of (Rule_at_True, N_L, N_j)=Counter_of (Rule_at_True, N_L, N_j)+1*

//Updating PS Score of N_L, N_j

//N_L attitude is related to the rule R_i

8. *PS (N_L, N_j)= Counter_of (Rule_at_True, N_L, N_j)/NE*
9. *End If*
10. *End For*
11. *End*

In other words, for a particular node L, $PS_{L,j}$ is a score that estimates the ratio of the veracity of an attitude between the node N_L and any given node N_j involved in the routing events observed by node N_L, averaged over the period extending from the first routing event observed at node N_L and involving node N_j until now. As we shall see later, these scores are used by the node to decide which entries to keep in the node forwarding table.

Since the RLA functionalities are based on events listened from the environment, it provides a dynamic size of the LKB (Local Knowledge Base), which is used in order to maintain a historical view of old events. A node L can estimate network stability by monitoring network variability using the rate of events R_L expressed in number of events per second. Then, we introduce a Stability Indicator (SI_L) metric estimated by the variance of the rate of events, observed over a fixed interval of time, dictated by the LKB size. The SI_L is computed using Equation (4) where n is the number of measurements made by node L of R_L. As a result, if the RLA detects a considerable increase (respectively decrease) of the SI_L then it increases (respectively decreases) the size of the LKB proportionally. If not, the size is left as it is.

$$SI_L = \frac{1}{n}\sum_{i=1}^{n}(R_{L,i} - \overline{R}_L)^2 \text{ , where } \overline{R}_L = \frac{1}{n}\sum_{i=1}^{n} R_{L,i} \qquad (4)$$

The controller has a PS Table which is used to maintain PS scores of all nodes known to the local node. At a final step, the controller sends a notification to the report creator which consults the PS table, and sends a message to the planner containing node identifiers concatenated to the associated PS score.

- Planner

It is responsible for the generation of action plans by using received information from the analyzer. It uses a coefficient called PST (Pairewise Score Threshold) in order to decide which action to do. Actions are decided as follows:

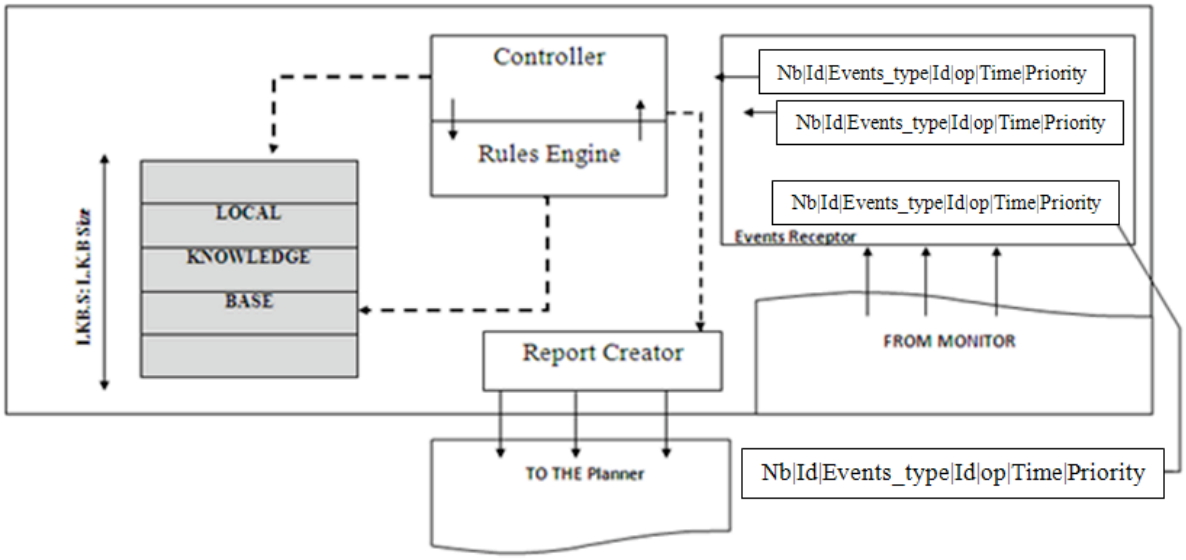

Fig. 12. Structure of an analyzer

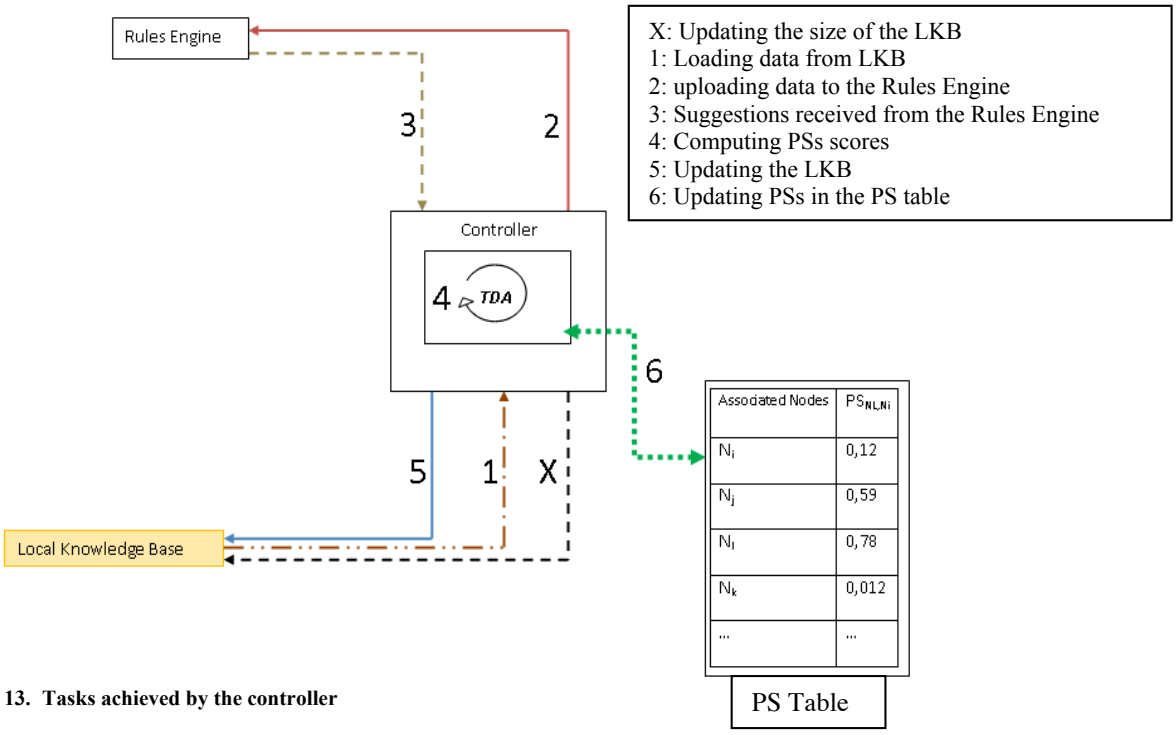

Fig. 13. Tasks achieved by the controller

1. *Let N_L :the Local node Id*
2. *Let N_j : a node known by N_L*
3. *Let $PS_{NL, Nj}$: Pairewise Score associated by N_L to N_j*
4. *For each $N_j \in$ Routing table*

// delete all routing table entries that are no longer needed

5. *if PS (N_L, N_j)≈0 then DEL_ENT (N_j)*

// suspend all routing tabl entries which present temporary problems

6. *if (0<<PS (N_L, N_j))<PST then SUSP_ENT(N_j)*

// update all entries

7. *if (PST≤PS (N_L, N_j)≤1 then ((UPD_ENT(N_j) OR exit))*
8. *if (PST≤PS (N_L, N_j)≤1 then ((UPD_ENT(N_j) OR exit))*
9. *End of each*

As a last step, the planner sends actions to the executor.

- Executor

The executor verifies plans proposed by the planner (synchronization in time, possible errors, etc.) and then it sends actions to the effector.

4.4 RLA Alternate Decision Module

As previously indicated, the RLA architecture exhibits general concepts of an autonomic architecture. Furthermore it integrates an Alternate Decisional Module (ADM) permitting the treatment of urgent cases (when immediate action must be taken and switching to the AM is not required). The ADM is crucial in highly dynamic networks such as wireless ad hoc networks. We can distinguish two such situations:

(1) A link is broken while that link is in use (events

received indicating an error involving nodes which serve as relays of an active route). The corresponding routing entry in the routing table has to be suspended until a next update.

(2) A destination node which is no more reachable (events received involving errors about a destination node in use, the transfer has to be stopped). The corresponding entry has to be deleted. Another entry to that destination can then be determined if it reappears.

The ADM intervenes with two types of actions:
DEL_ENT(Delete_from_routing_table) or
SUSP_ENT(Suspend_in_routing_Table).

The ADM has to use two interfaces (Figure 14): one IN (connected to the sensor) and one OUT (connected to the effector).

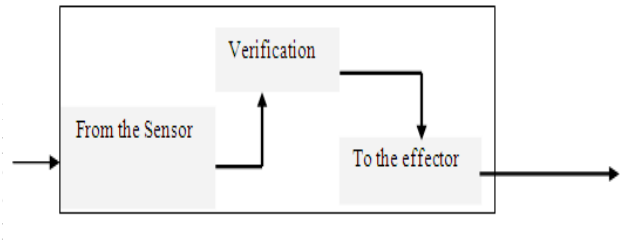

Fig. 14. Structure of an ADM

4.5 Communications between RLA agents

As aforementioned, updates performed by an RLA agent affect the routing behavior of the corresponding node. This change in behavior will be observed by the RLAs of other nodes based on monitored routing traffic. Consequently, if one RLA updates its managed resources (routing table), other RLAs will learn this from observed traffic and perform similar actions. In particular, nodes which are considered as cooperative have to be influenced by each other (if one node changes its behavior, the others inherit it).

4.6 The dynamic parameters PS & PST

PS should be viewed as a behavioral score, which captures the degree of positive interactions among pairs of nodes. Indeed, each time a node L receives a routing event from a node i, it triggers the verification of the six rules to see whether the observed event corresponds to a positive or negative interaction between nodes L and i.

PST (Pair-wise Score Threshold) is a parameter used by each node L to decide which node entries it should maintain active in its forwarding table. Only nodes with a PS > PST see their entries maintained active in node L forwarding table. PST is a dynamically estimated threshold, which is computed based on new received events (from the monitor) and recorded ones (in the LKB). PST is set equal to the ratio of the number of favorable events (NREs) to the total number of events, favorable (NRE) and unfavorable (RFEs) (Equation (5)).

Figure 15 illustrates an example of two nodes N_L and N_i which are cooperative, diligent or activator on a periodic basis (100 sec), i.e. collaborate positively (with respect to R_4, this corresponds to an increase of NRE events between the two nodes). Figure 15 shows how PS closely follows the same variations occurring among this pair of nodes. During the first 100 sec, N_L and N_i are mutually unknown, where PS values are nearly zero (about 0.001, 0.002). In the next period (100 sec to 200 sec), N_L and N_i become intensively cooperative, diligent and even activator, which explains the increased values of PS. The same mechanism is repeated periodically until 999sec. Figure 15 shows the variation of PS and PST over time:

1- In zones where N_L and N_i are cooperative (I: thick Dark gray in Figure 15), PS exceeds PST: entries containing N_i in the routing table of N_L are updated or left as they are.

2- In zones where N_L and N_i are moderately cooperative (II: uncolored in Figure 15, between 0.1 and 0.7), PS is inferior to PST (but quite different from 0), entries containing N_i in the routing table of N_L are suspended.

3- In zones where N_L and N_i are no more cooperative (III: light gray in Figure 15), PS ≈0, entries containing N_i in the routing table of N_L are deleted.

$$PST = \frac{NRE}{NRE + RFE} \quad (5)$$

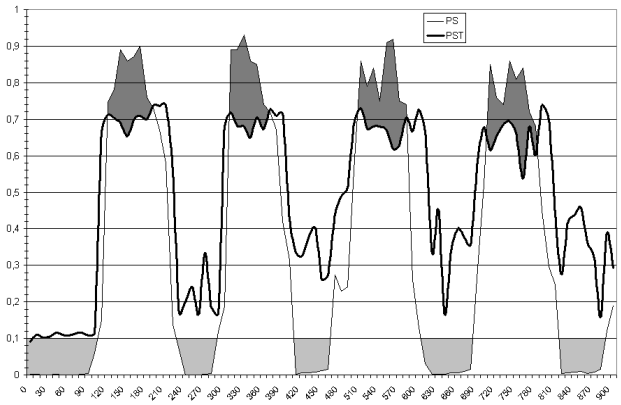

Fig. 15. Variations of PS and. PST over time. Dark gray presents Zones (I), light gray shows Zones (III), and the rest (uncolored) presents Zones (II). PST is presented by ''thick black Curve'' and PS by ''thin gray curve''. X-axis: PS, PST values, Y-axis: Time in seconds.

5. Experimental results

In this section, we describe experimental results showing how the proposed RLA-based autonomic architecture can help improve the operations of two well known routing protocols: the Ad-hoc On demand Distance Vector Routing (AODV) protocol [5], [15] and the Optimized Link State Routing (OLSR) protocol [4].

Experiments are conducted using NS2 simulator [14]. Simulations scenarios deploy a network composed of 100 nodes in a 1000m x 1000m field. Nodes are placed randomly on the surface and move according to the random way point mobility model with no pause time.

The simulations were run for 900 seconds with a number of generated connections varying between 15 and 20. For each connection, sources generate 512-byte data packets with a constant bit rate. In order to evaluate the proposed model, we have evaluated the most important performance metrics regarding the sensitivity to mobility, by varying the speed in the interval 0 to 20m/s.

-- Efficiency, in terms of delivered packet ratio

-- Overhead load: Control traffic generated compared to the data traffic delivered (in bits)

-- Average end-to-end delay: Time needed by a packet to reach the destination
-- Route stability: number of routes declared down per node.
-- Number of updates applied to the routing table.

a. Autonomic AODV (A²ODV)

AODV is a reactive protocol, where routes are established only when they are needed. AODV mechanisms are based on three types of requests: RREQ (Route Request), RREP (Route Reply) and RERR (Route Error). The route discovery mechanism uses a local broadcasting process; hence a receptor of a RREQ has to broadcast it to its neighbors with the requested destination. A receptor of a RREQ (an intermediate node or a destination) has to reply to the source only if it has a fresh route. After establishing a route between the source and destination, involved nodes are considered as active with respect to this route. Hence, each one has to make an update in its own routing table which helps in maintaining the route and detecting link errors.

For AODV, we have updated the ‘’aodv_routing’’ module which receives the PDU (Protocol Data Unit) from the application layer and loads the AODV algorithm. Updates made are reported in Table 4. For our specific need, a routing buffer is added at the third layer (serving as LKB), which could accumulate more than 64 data packets. This buffer allows the analyzer to treat events waiting for eventual processing.

Figure 16 and Figure 17 show a relatively low efficiency in terms of packet delivery ratio for both AODV and A²ODV. In low mobility case, failures and errors are caused by the MAC layer and not by the mobility. In that case, AODV has a unique option to apply: maintenance procedure (re-launching RREQ for new demanded routes and RERR for failures and errors). AODV drops all arriving packets to those destinations until new routes are established. However, A²ODV (RLA) uses storage data in the LKB to update the routing table, which enables it to retrieve lost destinations. Hence, routes are quickly recovered and packets will be delivered to their destinations. When nodes exhibit high mobility, A²ODV behaves practically like AODV. When failures are caused by mobility, the two versions converge to the same results.

As illustrated in Figure 18 and Figure 19, A²ODV results in a lower overhead than AODV. This is also confirmed by Figure 20, where the percentage of routes declared down for AODV is higher than A²ODV. This is due to:

- The mechanism used by AODV: ‘’using the route notification in the FIRST RREP received’’,
- The low number of delivered packets, which declare errors.
- A²ODV automatically adapts the content of its routing table based on monitored traffic history.

In Figure 21, we have used the term ‘’Mobility percentage’’ which corresponds to 0m/s (0%) and 20m/s (100%), as an average of speeds of all nodes in the network. Figure 21 clearly shows that ‘’shortest path routes” are not necessarily the best routes. A given long path can offer a better route in the long run than a shortest path. Figure 21 shows that A²ODV outperforms AODV in terms of average end-to-end delay, although AODV uses the shortest path (i.e. the first RREP received).

State	Signification	AODV	A²ODV
Init	Initialization	Yes	Same as AODV
Rcv_Appl	Received from upper layer	Yes	Same as AODV
Rcv_Mac	Received from lower layer	Yes	Same as AODV + RLA
Handle_RREQ_Rebroadcast	RREQ procedure	Yes	Same as AODV + RLA
Update_Route_Table	Updating routing table	Yes	Executed only by the RLA agent
Handle_link_failure	Link failure procedure	Yes	Same as AODV + RLA
RecvError	Received Error	Yes	Same as AODV + RLA
RecvReply	Received RREP	Yes	Same as AODV + RLA
RecvRequest	Received RREQ	Yes	Same as AODV + RLA
SendRequest	Sending RREQ	Yes	Same as AODV + RLA
rt_purge	Execute a control to the routing	Yes	Executed only by the RLA Agent
rt_resolve	Execute a local repair	Yes	Executed only by the RLA Agent
rt_down	shutdown a route	Yes	Executed only by the RLA Agent

TABLE 4. COMPARISON BETWEEN A²ODV VS AODV

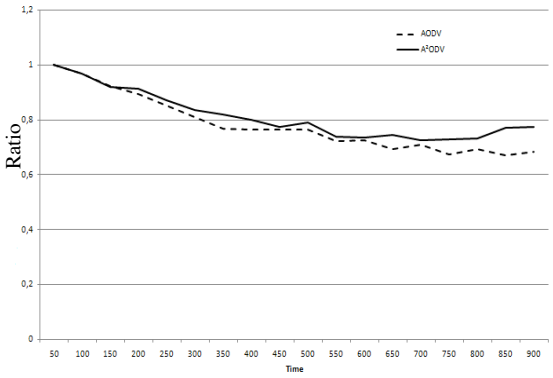

Fig. 16. Packet Delivery Ratio in Low Mobility (0m/sec≤speed≤4m/sec)

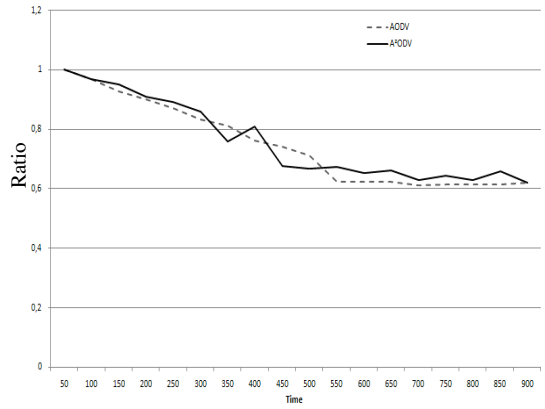

Fig. 17. Packet Delivery Ratio in high Mobility (5m/sec≤speed≤20m/sec)

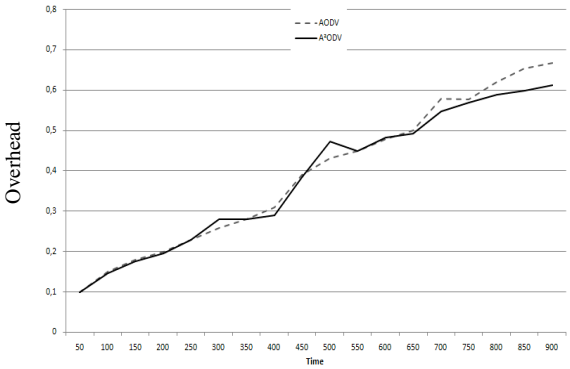

Fig. 18. Overhead (low mobility) (0m/sec≤speed≤4m/sec)

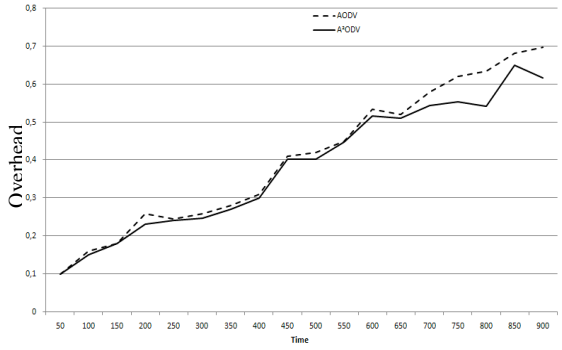

Fig. 19. Overhead (high mobility) (5m/sec≤speed≤20m/sec)

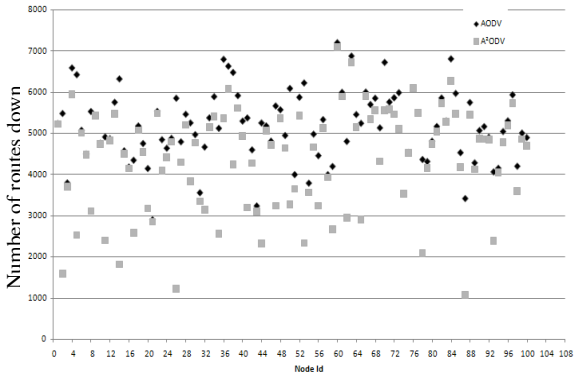

Fig. 20. Routes declared down, per node

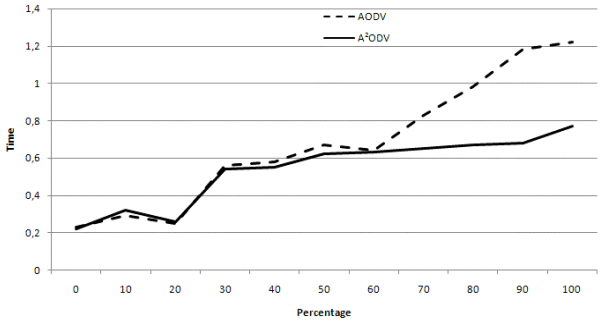

Fig. 21. Average path delay (sec) vs Mobility percentage

b. Autonomic OLSR (AOLSR)

OLSR [4] deploys three essential routing mechanisms: Hello messages, flooding control traffic using MPRs (Multi-Point Relays) and shortest path first algorithm. Hello messages are sent only for one hop and serve to discover neighbors, which are localized in the range of the local node. Links can be symmetric or asymmetric; OLSR considers that two nodes which are two-hop neighbor (via another node) can not behave as neighbor even if links are symmetric. The HELLO message contains a list of neighbors and the status of each corresponding link. For this reason, HELLO messages are considered as an immediate informative packet which updates the quality of links of the neighborhood of a given node. In addition, this information is to be used for a determined period and refreshed periodically. Thus, we can imagine the number of flooded HELLO packets across a large size MANET; which can engenders inacceptable control traffic. For this reason, OLSR provides MPR concept, where each node uses its two-hop neighbor list in order to determine a minimal set of MPRs. This guaranties that all neighbors at two hops remain reachable. Furthermore, each node has to maintain a list of nodes qualified as MPRs. MPRs are used in the flooding mechanism in order to reduce the flooding process and to periodically maintain the topology. The topology maintenance is based on TC messages (Topology Control), which contains the source address of the initiator node and its MPR selector set. Zach node has a partial view of the rest of the network, but using the MPRs, all nodes are reachable from each node of the network. We have to note that the topology Information is stored only for a given period of time and has to be updated periodically. For an autonomic implementation of OLSR, when a node receives a hello message containing a list of neighbors and the associated links status, it identifies the difference between the neighborhood information contained in such lists and the current links in use, and then (1) classifies each node as initiator of NRE (Positive Hello) or RFE (Negative Hello), (2) calculates the PS (respectively the PST) for each involved node, (3) updates the LKB and finally (4) applies the RE for new updates. Then, using PST threshold, the RLA decides whether or not the acquired changes have to be applied to the routing table (Managed Resource). The same mechanism is deployed after receiving MPRs notifications (via TC messages).

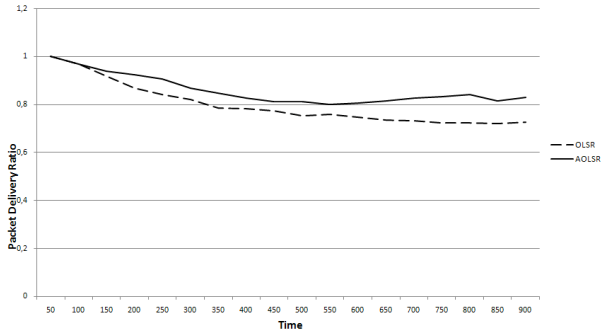

Fig. 22. Packet Delivery Ratio in Low Mobility (0m/sec≤speed≤4m/sec)

For OLSR, we have updated the *''kernel_routes''* module in order to integrate the RLA structure, which has to be exhibited for each operation. In the current version of AOLSR, the detection of needed routes is achieved at the application layer, which simplifies the task of the RLA agent. In addition, the LKB is modeled as a routing buffer at the third layer, and used by the two modules: *''kernel_routes''* and *''net_olsr''* and could accumulate more than 64 data packets. Furthermore, AOLSR deploys an RLA agent in each node, including the MPRs. For MPR nodes, simulations show that the number of updates made in LKB is more important compared to those of a normal node. This is entirely normal, because the quantity of the traffic transferred by MPR nodes is higher than the traffic forwarded by a normal node.

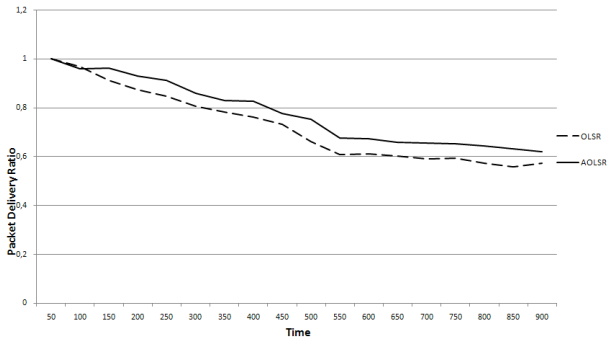

Fig. 23. Packet Delivery Ratio in high Mobility (5m/sec≤speed≤20m/sec)

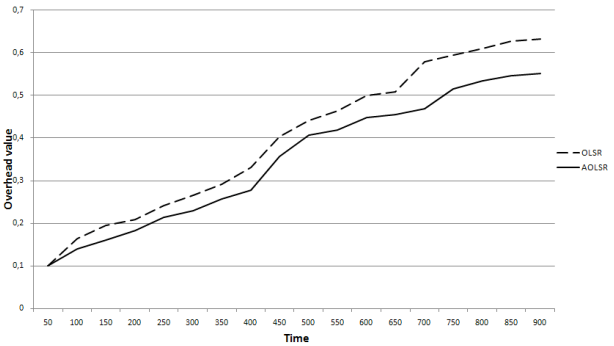

Fig. 24. Overhead (low mobility) (0m/sec≤speed≤4m/sec)

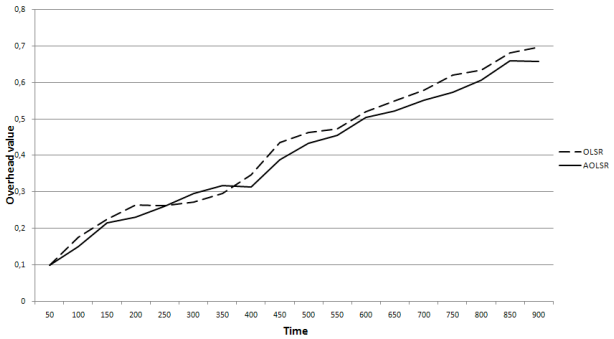

Fig. 25. Overhead (high mobility) (5m/sec≤speed≤20m/sec)

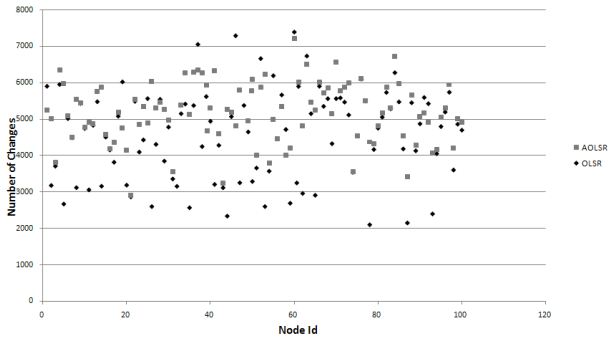

Fig. 26. Routes declared down, per node

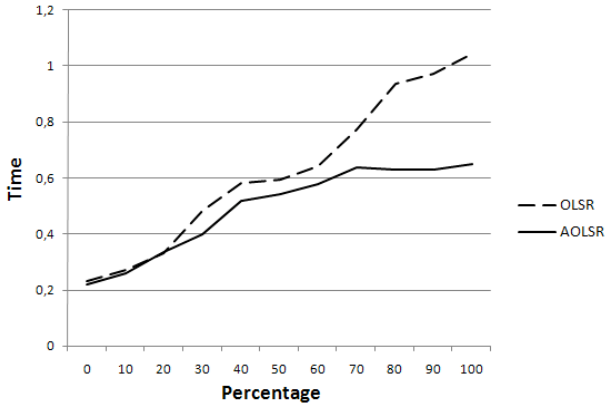

Fig. 27. Average path delay (sec) vs Mobility percentage

Figures 22 and 23 show that AOLSR performed noticeably better than OLSR with respect packet delivery ratio (not less than 83.2%). Note, that in low mobility, AOLSR behaves better than OLSR, AODV and A²ODV (78.1%). The reason is that AOLSR uses combined mechanisms: (1) information stored in the LKB, which involve intelligent decisions and (2) anticipates routing demands. In the case of high mobility, AOLSR achieves better results than OLSR. So, since changes are more frequent, then routing tables are reconfigured periodically. In that case, information about broken or partially known links is not considered in the routing tables, thus avoiding unstable routes. Furthermore, the route recovery mechanism is always accompanied by a loss of data packets. In the case of AOLSR, this problem is in a great part resolved, due to the nodes knowledge bases which allow a local recovery of routes based on background information filtered by the Rule Engine of the RLA agent. Figures 24, 25 and 26 show that the integration RLAs in the OLSR nodes results in less overhead. This is expected because the proactive properties of OLSR require periodic updates, in order to maintain topology structure throughout the entire network, which is avoided in AOLSR. Indeed, for AOLSR, when a route already exists in the routing buffer, AOLSR inhibits the rediscovery mechanism, which reduces significantly the overhead. As can be seen in Figure 27, AOLSR generates better routes than OLSR, especially for higher a mobility factor. In the beginning, the two protocols have nearly the same results until 26 seconds, after that, AOLSR surpasses OLSR.

7. Conclusion

This paper introduced an autonomic architecture, which is Self–adaptable, Self–manageable, Self-Configurable and Self-protectable. The architecture is designed to help better orchestrate node cooperation in carrying their various network activities (e.g. routing). The architecture is evaluated using the AODV and OLSR routing protocols as test cases. The architecture is deployed by an internal agent called RLA (Routing Learning Agent). Each mobile node has an RLA installed locally in order to dynamically self-adapt its routing decisions. An RLA is able to observe routing traffic, analyze it, and intervene on the local node routing table. Thus, the RLAs enable the corresponding network nodes self-adjust their routing decisions to changing network state. Extensive simulation results show that the proposed autonomic protocols (A²ODV, AOLSR) noticeably outperform the original AODV and OLSR with respect to traffic overhead, quality of routes, and packet delivery ratio.

References

[1] Anis Ben arbia, Habib youssef, ''Self-Organization in Wireless Networks using The Autonomic Behavior'', Second IEEE International workshop on ITS for ubiquitous ROADS, Hammamet, TUNISIA, May 2009.

[2] Duc A. Tran and Harish Raghavendra, "Congestion Adaptive Routing in Mobile Ad Hoc Networks", IEEE Transactions On Parallel And Distributed Systems, vol. 17, no. 11, NOVEMBER 2006, pp 1294-1305.

[3] S. Ziane and A. Mellouk "A Reinforcement Learning Approach for Routing and Scheduling Packets in Dynamic Networks". In proc of 1st International Conference on Information & Communication Technologies: from Theory to Applications, April 2004.

[4] T.H. Clausen, G. Hansen, L. Christensen and G.Behrmann "The Optimized Link State Routing Protocol, Evaluation through Experiments and Simulation." IEEE Symposium on "Wireless Personal Mobile Communications". September 2001.

[5] C. Perkins and E. M. Royer. Adhoc On demand Distance Vecto (AODV) routing. Draft-ietf-manet-aodv-02.txt, Nov 1998.

[6] A. Laouiti, A. Qayyum and L. Viennot "Multipoint Relaying for Flooding Broadcast Messages in Mobile Wireless Networks." 35th Annual Hawaii International Conference on System Sciences (HICSS'2002)

[7] A. J´osang and R. Ismail. ''The beta reputation system''. In 15th Bled Conference on Electronic Commerce, Bled, Slovenia, June 2002.

[8] A. J´osang, S. Hird, and E. Faccer. ''Simulating the effect of reputation systems on e-markets''. In Proceedings of the First International Conference on Trust Management, Crete, May 2003.

[9] S. Calomme, "Topologically-aware construction of unstructured overlays over ad hoc networks", Doctoral thesis, University of liege, ftp://ftp.run.montefiore.ulg.ac.be/pub/RUN-BK09-01.pdf

[10] Sungwon Kim, Chul-Ho Lee, Do Young Eun, "Super-Diffusive Behavior of Mobile Nodes and its Impact on Routing Protocol Performance," IEEE Transactions on Mobile Computing, 01 Jul. 2009.

[11] G. Di Caro and M. Dorigo, "AntNet: distributed stigmergetic control for communication networks". Journal of Artificial Intelligence Research, vol.9, pp. 317-365, 1998.

[12] Mouna Ayari, Zeinab Movahedi, Guy Pujolle, Farouk Kamoun: ADMA: autonomous decentralized management architecture for MANETs: a simple self-configuring case study. IWCMC 2009: pp132-137. 2009. http.dx.doi.org/10.1145/1582379.1582409

[13] Horn. Autonomic Computing: IBM's perspective on the State of Information Technology. (2001). IBM Corporation available at http://researchweb.watson.ibm.com/autonomic/manifesto/autonomic_computing.pdf

[14] "The network simulator – NS-2". Available online at http://www.isi.edu/nsnam/ns/ (page accessed on October 2009).

[15] C. Perkins, Nokia Research Center, E. Belding-Royer, University of California, Santa Barbara, Request For Comments 3561, July 2003.

[16] Wen Yan Wei Guo Jun Liu, ''An Implementation and Study of OLSR Protocol in Linux OS'', 4th International Conference on Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08: pp 1-4. 2008.

[17] IBM Corporation, An Architectural Blueprint for Autonomic Computing, available at http://www-306.ibm.com/autonomic/library.shtml